Title	A Short Primer on the Primal-Dual Method for Approximation Algorithms (Algorithm Engineering as a New Paradigm)
Author(s)	Williamson, David P.
Citation	数理解析研究所講究録 (2001), 1185: 221-230
Issue Date	2001-01
URL	http://hdl.handle.net/2433/64621
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
1185巻 2001年 221-230

A Short Primer on the Primal-Dual Method for Approximation Algorithms

David P. WILLIAMSON

IBM Almaden Research Center
650 Harry Rd., San Jose, CA, 95120, USA
dpw@almaden.ibm.com

Abstract: In this primer, we give an overview of a technique used to design and analyze algorithms that provide approximate solutions to NP-hard problems in combinatorial optimization. Because of parallels with the primal-dual method commonly used in combinatorial optimization, we call it the primal-dual method for approximation algorithms. We give a general overview of this technique and show how it can be used to derive approximation algorithms for a number of different problems, including a network design problem and a feedback vertex set problem.

Keywords: approximation algorithms, primal-dual method, Steiner trees, feedback vertex sets

1 Introduction

Many problems of interest in combinatorial optimization are considered unlikely to have efficient algorithms; most of these problems are NP-hard, and unless $P = NP$ they do not have polynomial-time algorithms to find an optimal solution. Researchers in combinatorial optimization have considered several approaches to deal with NP-hard problems. One approach that has received considerable attention recently is that of *approximation algorithms*. An *α-approximation algorithm* for an optimization problem is an algorithm that runs in polynomial time and produces a solution whose value is within a factor of α of the value of an optimal solution. The parameter α is called the *performance guarantee* or the *approximation ratio* of the algorithm. We will follow the convention that $\alpha \geq 1$ for minimization problems and $\alpha \leq 1$ for maximization problems, so that a 2-approximation algorithm for a minimization problem produces a solution of value no more than twice the optimal value, and a $\frac{1}{2}$-approximation algorithm for a maximization problem produces a solution of value at least half the optimal value. The reciprocal $1/\alpha$ is sometimes used in the literature for maximization problems, so that the examples above would both be referred to as 2-approximation algorithms.

In the past dozen years there have been a number of exciting developments in the area of approximation algorithms. For more details about the area of approximation algorithms, the reader is invited to consult the excellent survey of Shmoys [26], the book of surveys edited by Hochbaum [19], or the forthcoming monograph of Vazirani [29]. In this primer we will focus on one very useful algorithmic technique, called the primal-dual method for approximation algorithms, that has been developed and applied to several different problems in combinatorial optimization.

In order to discuss this method, it is necessary to have some familiarity with the theory of integer and linear programming. Good introductions can be found in Chvátal [6] or Strang [27, Ch. 8]. We give a very basic introduction here. In a *linear program* (LP), we try to minimize (or maximize) a linear *objective function* in variables $x_1, \ldots, x_n$ subject to linear constraints on the x_i. For example, consider the linear program

$$
\begin{aligned}
& \text{Min} && \sum_{i=1}^{n} c_i x_i && \\
& \text{subject to:} && && \\
(P) \quad & && \sum_{i=1}^{n} a_{ij} x_i \geq b_j && j = 1, \ldots, m \\
& && x_i \geq 0 && i = 1, \ldots, n.
\end{aligned}
$$

A particular setting of the x_i which obeys all the linear constraints is said to be a *feasible* solution. A setting of the x_i which minimizes the objective function is called an *optimal* solution. Associated with every linear program is a *dual* linear program. For instance, the dual of the linear program (P) above is

$$
\begin{aligned}
& \text{Max} \quad \sum_{j=1}^{m} b_j y_j \\
& \text{subject to:} \\
(D) \qquad & \sum_{j=1}^{m} a_{ij} y_j \leq c_i \qquad i = 1, \ldots, n \\
& y_j \geq 0 \qquad\qquad\quad j = 1, \ldots, m.
\end{aligned}
$$

This linear program (D) is the dual of (P), which is sometimes called the *primal* LP. The dual has many important properties, one of which is the following: the value of the dual objective function for any feasible dual y is a lower bound on the value of an optimal solution to the primal LP. Notice that there is a dual constraint i for each primal variable x_i and a primal constraint j for each dual variable y_j. Given a primal feasible solution x and a dual feasible solution y, the solution x is said to obey the primal *complementary slackness* conditions with respect to y if whenever $x_i > 0$ the corresponding dual constraint is met with equality by y. Similarly, y is said to obey the dual complementary slackness conditions with respect to x if whenever $y_j > 0$ the corresponding primal constraint is met with equality by x. If x and y obey both primal and dual complementary slackness conditions, it can be shown that they are optimal primal and dual solutions. Sometimes we add constraints to linear programs asking that x_i be a nonnegative integer or restricted to some bounded range of integers (e.g. $x_i \in \{0, 1\}$). A linear program with this type of constraint is usually called an *integer program*. Given an integer program, it is often useful to consider its *linear programming relaxation*; that is, the LP obtained by dropping the condition that the variable x_i be a nonnegative integer, and replacing it with the condition that $x_i \geq 0$. Observe that the value of an optimal solution for the linear programming relaxation will be no greater than the value of the integer program (in which we minimize the objective function) since any x feasible for the integer program will also be feasible for the relaxation, and will have the same objective function value.

The name "the primal-dual method" has been borrowed from a standard tool used in designing algorithms for polynomial-time solvable problems in combinatorial optimization. A good overview can be found in the textbook of Papadimitriou and Steiglitz [23] (see also the survey of Goemans and Williamson [16]). The primal-dual method for approximation algorithms is a simple modification of the standard method. Given a combinatorial optimization problem that can be formulated as an integer programming problem, the primal-dual method for approximation algorithms considers this primal integer program and a dual of a linear programming relaxation of the integer program. We start with a dual feasible solution y, and attempt to find a feasible integer primal solution x that obeys the primal complementary slackness conditions with respect to y. If one exists, we stop, otherwise we can show that we can modify y so as to increase the dual objective function value. In the standard primal-dual method, we would have wanted x and y that obeyed both primal and dual complementary slackness conditions, implying that both x and y are optimal. However, in general we can't expect that there exists an optimal integral solution to the linear programming relaxation, so we drop the dual complementary slackness conditions.

As we will see below, relaxing the dual complementary slackness condition in appropriate ways leads to provably good algorithms for NP-hard problems in combinatorial optimization by yielding solutions to the primal integer problem that cost no more than α times the value of a feasible solution to the dual, which implies that the solution is within a factor of α of optimal. The value of the dual solution is always within some factor of α of optimal, but may from instance to instance be much closer; since we generate a dual each time we can know when we are closer than factor of α from the optimum.

In the next section, we develop the basic ideas given above into a primal-dual algorithm for a generic problem, and give theorems for its analysis. We conclude in Section 3.

Other surveys on the primal-dual method have been given by Goemans and Williamson [16] and

Bertsimas and Teo [4] (see also the thesis of Teo [28]). Our central exposition is taken from a forthcoming survey of Williamson [31] and closely follows that of [16].

2 The primal-dual method for approximation algorithms

2.1 The hitting set problem

We now show how the primal-dual method can be used to give approximation algorithms for NP-hard problems in combinatorial optimization. In order to do this, it will be useful to consider the *hitting set problem*: given a ground set of elements E, nonnegative costs c_e for all elements $e \in E$, and subsets $T_1, \ldots, T_p \subseteq E$, we want to find a minimum-cost subset $A \subseteq E$ so that A has a nonempty intersection with each subset T_i. We say that A *hits* each subset T_i.

The hitting set problem can be used to model a number of NP-hard problems, and we will consider several in this section. In the *minimum-weight vertex cover problem*, we are given a graph $G = (V, E)$ with weights $w_i \geq 0$ for all vertices $i \in V$, and we must select a minimum-weight subset of vertices such that each edge is covered (that is, at least one of its endpoints is chosen). We can formulate the minimum-weight vertex cover problem as a hitting set problem in which the ground set elements are vertices, and we have a subset $T_i = \{u, v\}$ for each edge (u, v) in the graph. In the *minimum-weight feedback vertex set problem in undirected graphs*, we are given as input an undirected graph $G = (V, E)$ and nonnegative weights $w_i \geq 0$ on the vertices $i \in V$, and the goal is to remove a minimum-weight set of vertices from G so as to make the remaining graph acyclic. We can view this as a hitting set problem in which the ground set elements are the vertices of the graph, and we must hit every cycle in the graph; that is, $T_i = C_i$, where C_i is the ith cycle of G. In the *shortest s-t path problem*, we are given an undirected graph with nonnegative edge costs c_e for all $e \in E$, and two distinguished vertices s and t, and we must find the minimum-cost path from s to t. We can formulate this as a hitting set problem in which the edges are the ground set elements and we must hit every cut in the graph separating s from t; that is, for all $S_i \subseteq V$ with $s \in S_i$ and $t \notin S_i$, we must select an edge from $T_i = \delta(S_i)$, where $\delta(S)$ is the set of edges with exactly one endpoint in S. By the max-flow/min-cut theorem of Ford and Fulkerson [12], we have selected an edge in every cut separating s from t iff there is a path from s to t. In the *minimum-cost branching problem* we are given a directed graph $G = (V, A)$, nonnegative costs c_a for all arcs $a \in A$, and a root vertex $r \in V$, and the goal is to find a minimum-cost branching (a set of arcs such that for every vertex, there is a path from the root to the vertex). By using a max-flow/min-cut argument, one can see that the following hitting set problem models the minimum-cost branching problem: the ground set of elements are the arcs, and for every set of vertices $S_i \subseteq V - r$, we must hit the set $\delta^-(S_i)$ of arcs, where $\delta^-(S_i)$ is the set of arcs whose heads are in S_i and tails are not in S_i. Finally, in the *generalized Steiner tree problem* we are given an undirected graph $G = (V, E)$, nonnegative costs $c_e \geq 0$ on all edges $e \in E$, and k pairs of vertices $s_j, t_j \in V$. The goal is to find a minimum-cost set of edges F, such that for each $j = 1, \ldots, k$, s_j and t_j are connected in the graph (V, F). Again, a max-flow/min-cut argument will show that the problem can be modelled by the hitting set problem in which the ground set elements are the edges and we must hit every cut that separates some s_j-t_j pair; in other words, for each S_i such that for some j, $|S_i \cap \{s_j, t_j\}| = 1$, we must hit $T_i = \delta(S_i)$.

Except for the minimum-cost s-t path problem and the minimum-cost branching problem, all of the problems above are NP-hard. For many of them, the size of the hitting set formulation is exponential in the size of the input. For example, in the feedback vertex set problem, the number of cycles can be exponential in the size of the graph. We will see later that the primal-dual method can often be used in this case and still result in a polynomial-time algorithm.

We can model the hitting set problem by the following integer program:

$$\begin{aligned} \text{Min} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to:} \quad & \\ & \sum_{e \in T_i} x_e \geq 1 \qquad \forall i \\ & x_e \in \{0, 1\}. \end{aligned}$$

If we relax the integrality constraint $x_e \in \{0,1\}$ to $x_e \geq 0$ and take the dual of the resulting linear program, we obtain the following:

$$\text{Max} \quad \sum_i y_i$$

subject to:

$$\sum_{i:e\in T_i} y_i \leq c_e \qquad \forall e \in E$$

$$y_i \geq 0 \qquad \forall i.$$

Our goal is to construct a feasible solution $\bar{x}$ to the primal integer program and a feasible solution y the dual linear program such that $\sum_{e\in E} c_e \bar{x}_e \leq \alpha \cdot \sum_{i=1}^{p} y_i$ for some value of α. This implies that the cost of our primal solution is no more than α times the cost of an optimal solution to the integer program. If we can construct our solutions in polynomial time, then we have an α-approximation algorithm. We will sometimes give our primal solution as $\bar{x}$ or as a subset $A \subseteq E$, which implies the solution $\bar{x}_e = 1$ for $e \in A$ and $\bar{x}_e = 0$ otherwise.

2.2 The basic algorithm

Let's consider how to develop a primal-dual algorithm for the hitting set problem based on what we have said so far. Suppose we are given a dual solution y. We maintain the primal complementary slackness conditions; that is, we include e in our solution only if the corresponding dual inequality is *tight* (that is, met with equality). Thus $e \in A$ implies $\sum_{i:e\in T_i} y_i = c_e$. Suppose that we simply set A to contain all elements e for which the corresponding dual inequality is tight; that is, $A = \{e \in E : \sum_{i:e\in T_i} y_i = c_e\}$. According to the primal-dual method, if A is not a feasible solution to the hitting set problem for our current dual solution y, then there must be a way to modify y that increases the dual objective function value. In the case of the hitting set problem, if A is not feasible, then there must be some set T_k such that $A \cap T_k = \emptyset$. We call such a set T_k a *violated* set. If T_k is violated, then for all $e \in T_k$, their corresponding dual inequalities must not be tight; that is, $\sum_{i:e\in T_i} y_i < c_e$. However, since these inequalities are the only ones in which y_k appears, we can increase y_k until some dual constraint becomes tight for some $e \in T_k$, and obtain another dual feasible solution which now has a larger objective function value. Furthermore, since a new constraint became tight for some $e \in T_k$, we can add this element to A, and now $A \cap T_k \neq \emptyset$. This leads to the basic primal-dual algorithm, which is shown in Figure 1. This algorithm is due to Bar-Yehuda and Even [2].

We now analyze the cost of the feasible solution A returned by the algorithm. The cost of A is

$$\sum_{e\in A} c_e = \sum_{e\in A} \sum_{i:e\in T_i} y_i \qquad (1)$$

$$= \sum_{i=1}^{p} y_i |A \cap T_i|, \qquad (2)$$

where (1) follows since the complementary slackness conditions are obeyed for the primal variables, and (2) follows by reversing the double sum. If we let $f = \max_i |T_i|$, then certainly $|A \cap T_i| \leq f$ for all i, so that

$$\sum_{e\in A} c_e \leq f \cdot \sum_{i=1}^{p} y_i.$$

We thus have an f-approximation algorithm for the hitting set problem, since the dual objective function value is a lower bound on the cost of an optimal solution to the hitting set problem. As an example of what can be proved in this case, recall that for the minimum-weight vertex cover problem each subset T_i contained the two endpoints of an edge in a graph, so that $|T_i| = 2$ for each i in this case. Thus the algorithm gives a 2-approximation algorithm for the minimum-weight vertex cover problem.

2.3 Feedback vertex sets

We now turn to a slightly more complicated application of the primal-dual algorithm: the feedback vertex set problem for undirected graphs. Let A and y be the primal and dual solution created by the algorithm. Recall from equations (1) and (2) if for any $y_i > 0$ it is the case that $|A \cap T_i| \leq \alpha$, then the algorithm is an α-approximation algorithm. Recall now that for the hitting set problem modelling this problem, each ground element e is a vertex j, the cost c_e is the vertex weight w_j, and the sets T_i are the cycles in the graph. In this case, Bar-Yehuda, Naor, Geiger, and Roth [3] obtain a performance guarantee of $4 \log_2 n$ (where $n = |V|$) by carefully choosing the cycle in line 4 of the algorithm, and by noticing that one can

```
1   y ← 0
2   A ← ∅
3   While A is not feasible
4       Find violated T_k (i.e. T_k s.t. A ∩ T_k = ∅)
5       Increase y_k until ∃ e ∈ T_k such that Σ_{i:e∈T_i} y_i = c_e
6       A ← A ∪ {e}
7   Return A.
```

Figure 1: The basic primal-dual algorithm.

succesfully ignore some vertices since their corresponding dual inequalities will always be satisfied. In order to choose the violated cycle, Bar-Yehuda et al. invoke the following lemma of Erdős and Pósa [10].

Lemma 1 (Erdős and Pósa [10]) *Given a graph $G' = (V', E')$ with no degree 1 vertices and with every vertex of degree 2 adjacent to two vertices of higher degree, there exists a cycle of length no longer than $4\log_2|V'|$, and it can be found in polynomial time.*

Of course, the given input graph might not meet the conditions of the lemma. Thus we show that we can ignore some vertices; the remaining vertices we call *special* vertices. We map the graph onto a graph G' that contains exactly the special vertices, such that there is a bijective mapping between cycles of G and of G'. Then by applying the lemma we can find in G a violated cycle of at most $4\log_2 n$ special vertices, and since we only add special vertices to A, we get that for any $y_i > 0$ (corresponding to some violated cycle T_i chosen in line 4), $|A \cap T_i| \leq 4\log_2 n$, implying the desired performance guarantee.

Now we need to specify which vertices we can ignore, and why their dual inequalities will remain feasible. Suppose that as we add a vertex j to A in line 6 of the algorithm, we remove j and its incident edges from the graph. Certainly we can ignore any vertex in the remaining graph that is no longer in a cycle; since we only add vertices from the chosen violated set (line 5), we only add vertices that are in some cycle. Now consider any path of vertices that all have degree 2. Since any cycle that goes through one of these vertices must go through all of them, it must be the case that when the reduced cost $\tilde{w}_j = w_j - \sum_{i:j\in T_i} y_i$ of a vertex j on this path decreases by ϵ, the reduced cost of all vertices on this path also decreases by ϵ. Thus we can safely ignore all vertices in this path except for one special vertex j with the smallest reduced cost, since no dual inequality for any vertex on the path will become tight unless the dual inequality for j becomes tight. Furthermore, if j is added to A, then all cycles containing the vertices on this path will be hit, and so no other vertex from the path need by added to A.

Since we can ignore any vertex not on a cycle, and ignore all but one vertex on a path of vertices of degree 2, we obtain the desired graph G' from G by removing all vertices currently in A, recursively removing all degree 1 vertices, and replacing any path of degree 2 vertices with the special vertex for that path. This yields a graph G' obeying the properties of the lemma, such that any cycle in G' has a one-to-one mapping to a cycle of G. Thus we can find a cycle of at most $4\log_2 n$ special vertices in G.

This argument yields a $(4\log_2 n)$-approximation algorithm for the minimum-weight feedback vertex set problem in undirected graphs. In fact, one can obtain a 2-approximation algorithm for this problem using the primal-dual method, but one must use a different integer programming formulation of the problem; see Chudak et al. [5] and Fujito [13] for details.

2.4 Reverse delete

We now turn to modifications of the basic primal-dual algorithm of Bar-Yehuda and Even. The first is a relatively simple idea: once a feasible solution A has been obtained, we should exam-

ine the elements of A and delete any that are not needed for a feasible solution. This idea was first introduced by Goemans and Williamson [15], but we will present here a refinement discovered independently by Klein and Ravi [21] and Saran, Vazirani, and Young [25]. They showed that it is useful for the analysis of the algorithm to examine the elements of A for possible deletion in a certain order; in particular, in the reverse of the order in which the elements of A were added. This part of the algorithm is sometimes called the *reverse delete*. We present the modified algorithm in Figure 2.

To see why the reverse delete step is useful for the analysis, consider the set T_{i_l} chosen in the lth iteration of the algorithm. Let A_l be the set of elements in A at the beginning of the lth iteration, let e_l be the element added in the lth iteration, and let A' be the final set returned by the algorithm. By the analysis at the beginning of the section (Equations (1) and (2)), if we can show that $|A' \cap T_{i_l}| \leq \alpha$ for all iterations l, we have an α-approximation algorithm. Note that since T_{i_l} is chosen as a violated set, it is the case that $T_{i_l} \cap A_l = \emptyset$, so if $B = A' - A_l$, then we only need prove that $|B \cap T_{i_l}| \leq \alpha$. Furthermore, when e_l is considered for deletion, no element e_j for $j < l$ has been considered for deletion, so the contents of A at that point in time in the reverse delete step must be precisely $A_l \cup B$. Finally, because each element in B was added after the lth iteration, it must be the case that each of them was already considered by the reverse delete step and is necessary for the feasibility of $A_l \cup B$. Thus for any $e \in B$, $A_l \cup B - e$ is not a feasible solution. We call any set of elements D such that $A_l \cup D$ is feasible an *augmentation* of A_l, and any augmentation D such that for any $e \in D$, $A_l \cup D - e$ is not feasible, a *minimal* augmentation. We have shown above that B is a minimal augmentation of A_l. We are trying to bound $|B \cap T_{i_l}|$, and certainly this is dominated by the maximum of $|D \cap T_{i_l}|$ over all minimal augmentations D of A_l. Thus we have shown the following theorem.

Theorem 2 *If for all iterations l of the algorithm in Figure 2,*

$$\max_{D: \text{min. aug. of } A_l} |D \cap T_{i_l}| \leq \alpha,$$

the algorithm is an α-approximation algorithm.

To illustrate the use of this analysis, we consider the shortest s-t path problem and the minimum-cost branching problem. Recall that for the minimum-cost s-t path problem, we need to hit the sets $T_i = \delta(S_i)$ for all sets S_i with $s \in S_i$, $t \notin S_i$, where $\delta(S_i)$ is the set of edges with exactly one endpoint in S_i. To apply the primal-dual algorithm of Figure 2, we need to specify which violated set T_{i_l} is chosen for a given infeasible solution A_l. Here we invoke a principle that turns out to be useful for a number of problems of this sort: we choose the *minimal* violated set $T_i = \delta(S_i)$, where by this we mean a set S_i such that there is no other set $S_j \subset S_i$ with $T_j = \delta(S_j)$ also violated. For the minimum-cost s-t path problem, this principle implies that for an infeasible solution A_l, we find the connected component S_{i_l} containing s in the graph (V, A_l), and choose the violated set $T_{i_l} = \delta(S_{i_l})$. It is not difficult to see that for any augmentation D of A_l, if $|D \cap \delta(S_{i_l})| > 1$, then an edge of $D \cap \delta(S_{i_l})$ can be removed with the remaining edges still containing an s-t path. Thus for any minimal augmentation D, it is the case that $|D \cap \delta(S_{i_l})| = 1$, which implies by the analysis of the preceding paragraph that the primal-dual method gives a 1-approximation algorithm, or an optimal algorithm, for the shortest s-t path problem. In fact, one can show that this algorithm is just Dijsktra's algorithm [8, 30].

For the minimum-cost branching problem, we need to hit the sets $T_i = \delta^-(S_i)$ for all $S_i \subseteq V - r$. Recall that $\delta^-(S_i)$ is the set of arcs with their heads in S_i and their tails not in S_i. Given an infeasible set A_l, we find a strongly connected component S_{i_l} not containing the root r for which $A \cap \delta^-(S_i) = \emptyset$ and choose as our violated set $T_{i_l} = \delta^-(S_{i_l})$. It is not hard to show that such a strongly connected component must exist if A_l is infeasible. Then again it is easy to see that for any augmentation D of A_l, only one arc in $D \cap \delta^-(S_{i_l})$ is necessary, since the strong connectivity of S_{i_l} implies all the vertices of S_{i_l} can be reached through that arc. Hence for any minimal augmentation D of A_l, $|D \cap \delta^-(S_{i_l})| = 1$, and we again have an optimal algorithm. One can show that this algorithm is the same as Edmonds' algorithm for the minimum-cost branching problem [9].

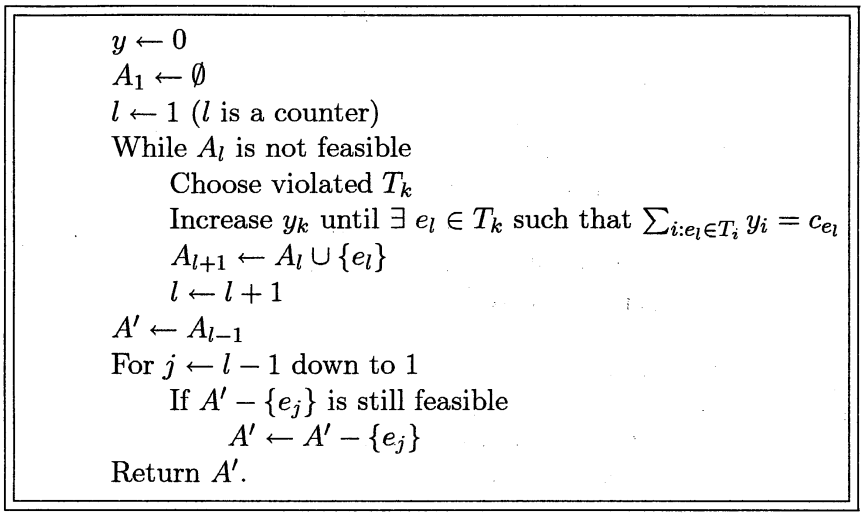

Figure 2: The primal-dual algorithm with reverse delete step added.

2.5 Increasing multiple duals

We now introduce another modification to our primal-dual algorithm. To motivate the modification, we consider the generalized Steiner tree problem. Recall that this can be modelled by a hitting set problem in which we must hit all $T_i = \delta(S_i)$ such that $|S_i \cap \{s_j, t_j\}| = 1$ for some s_j-t_j pair that must be connected. Suppose we try to apply the algorithm in Figure 2 and the analysis above to this problem. As with the shortest s-t path problem, we will invoke the principle of finding a minimal violated set and choose some connected component S_{i_l} of (V, A_l) such that $|S_{i_l} \cap \{s_j, t_j\}| = 1$, and choose as our violated set $T_{i_l} = \delta(S_{i_l})$. However, consider the problem in which $s = s_1 = s_2 = \cdots = s_k$, and $t_1, \ldots, t_k$ are distinct vertices. Then for $A_1 = \emptyset$, the vertex s and each t_j is a possible minimal violated set. Without loss of generality, suppose we choose the violated set $T = \delta(\{s\})$. Then one possible minimal augmentation is $D = \{(s, t_1), (s, t_2), \ldots, (s, t_k)\}$, and $|D \cap T| = k$. Thus the algorithm and analysis we have developed so far would only give a k-approximation algorithm.

However, if we consider the number of times this augmentation hits these minimal violated sets averaged over the number of minimal violated sets, we get something better: $|D \cap \delta(\{s\})| = k$, but $|D \cap \delta(\{t_j\})| = 1$, with $k+1$ minimal violated sets, leading to an average of $2k/(k+1) \approx 2$. This leads to the following idea: suppose we choose multiple violated sets and increase their dual variables simultaneously and uniformly. It turns out that this gives good approximation algorithms for a number of problems, including a 2-approximation algorithm for the generalized Steiner tree problem. We give the modified algorithm in Figure 3. The idea of increasing multiple duals was introduced implicitly by Agrawal, Klein, and Ravi [1] (who did not refer to LP duality), and was made explicit by Goemans and Williamson [15].

We now show how we can analyze the algorithm in Figure 3 via the following theorem. Notice that this algorithm and its analysis generalize the algorithm of Figure 2, in which only one violated set is chosen in each iteration.

Theorem 3 *If for every iteration l of the algorithm in Figure 3,*

$$\max_{D:\text{min. aug. of } A_l} \sum_{T_k \in \mathcal{V}_l} |D \cap T_k| \leq \alpha |\mathcal{V}_l|,$$

the algorithm is an α-approximation algorithm.

Proof: Let A' be the final solution returned by the algorithm. We wish to prove that $\sum_{e \in A'} c_e \leq \alpha \sum_{i=1}^{p} y_i$. As before, we have that

$$\sum_{e \in A'} c_e = \sum_{e \in A'} \sum_{i: e \in T_i} y_i = \sum_{i=1}^{p} |A' \cap T_i| y_i.$$

$y \leftarrow 0$
$A_1 \leftarrow \emptyset$
$l \leftarrow 1$ (l is a counter)
While A_l is not feasible
    Choose set $\mathcal{V}_l$ of violated sets
    Increase y_k uniformly for all $T_k \in \mathcal{V}$ until $\exists\ e_l \notin A_l$
        such that $\sum_{i:e_l \in T_i} y_i = c_{e_l}$
    $A_{l+1} \leftarrow A_l \cup \{e_l\}$
    $l \leftarrow l + 1$
$A' \leftarrow A_{l-1}$
For $j \leftarrow l - 1$ down to 1
    If $A' - \{e_j\}$ is still feasible
        $A' \leftarrow A' - \{e_j\}$
Return A'.

Figure 3: The general primal-dual algorithm.

So we need to prove that

$$\sum_{i=1}^{p} |A' \cap T_i| y_i \leq \alpha \sum_{i=1}^{p} y_i.$$

Let ϵ_l be the amount by which the duals are increased in iteration l of the algorithm. Then clearly, for the solution y at the end of the algorithm,

$$\sum_{i=1}^{p} y_i = \sum_{l} |\mathcal{V}_l| \epsilon_l.$$

Similarly,

$$\begin{aligned} \sum_{i=1}^{p} |A' \cap T_i| y_i &= \sum_{i=1}^{p} |A' \cap T_i| \sum_{l:T_i \in \mathcal{V}_l} \epsilon_l \\ &= \sum_{l} \left(\sum_{T_k \in \mathcal{V}_l} |A' \cap T_k| \right) \epsilon_l. \end{aligned}$$

Thus certainly the inequality follows if for all iterations l,

$$\sum_{T_k \in \mathcal{V}_l} |A' \cap T_k| \leq \alpha |\mathcal{V}_l|.$$

As in the proof of Theorem 2, $\sum_{T_k \in \mathcal{V}_l} |A' \cap T_k|$ is dominated by $\max_{D:\text{min. aug. of } A_l} \sum_{T_k \in \mathcal{V}_l} |D \cap T_k|$. Thus the theorem follows. □

To illustrate the use of the algorithm and the theorem, we show how we can obtain a 2-approximation algorithm for the generalized Steiner tree problem. As suggested above, in each iteration l we choose all the minimal violated sets; that is, we choose the sets $T_i = \delta(S_i)$ for all connected components S_i in (V, A_l) such that for some j, S_i contains exactly one of s_j or t_j. Thus $\mathcal{V}_l$ is the set of all such sets T_i.

Theorem 4 *Using the algorithm in Figure 3 with the choice of $\mathcal{V}_l$ as given above yields a 2-approximation algorithm for the generalized Steiner tree problem.*

Proof: To prove this, we show that the statement of Theorem 3 holds for $\alpha = 2$. To do this, we consider the graph in which each connected component of (V, A_l) has been shrunk to a single node; let V' be this set of vertices. Let D be any minimal augmentation of A_l, and consider the graph $H = (V', D)$. Note first that H is a forest, otherwise D is not minimal. Observe also that some of the vertices in V' correspond to connected components S_i that are in $\mathcal{V}_l$ and some do not. Let $R \subseteq V'$ be the first type of vertex, which we will call red, and $B = V' - R$ be the second type, which we will call blue. Observe that $|R| = |\mathcal{V}_l|$. Also, if $deg(v)$ is the degree of $v \in V'$ in the graph H, and v corresponds to the connected component S_i in (V, A_l), then $|D \cap T_i| = |D \cap \delta(S_i)| = deg(v)$. Thus the desired

inequality

$$\sum_{T_k \in \mathcal{V}_l} |D \cap T_k| \leq 2|\mathcal{V}_l|$$

reduces to proving that $\sum_{v \in R} deg(v) \leq 2|R|$. If we can show that no blue vertex has degree 1, then the statement would follow, since (ignoring blue vertices of degree 0),

$$\begin{aligned} \sum_{v \in R} deg(v) &= \sum_{v \in R \cup B} deg(v) - \sum_{v \in B} deg(v) \\ &\leq 2(|R| + |B|) - 2|B| \\ &\leq 2|R|. \end{aligned}$$

The inequalities follow since the sum of degrees of the vertices in the forest H is no more than twice the number of vertices, and every blue vertex in the sum has degree at least 2. To show that no blue vertex has degree 1, assume the opposite: let v be a blue vertex of degree 1, let $e \in D$ be the adjacent edge in H, and let S be the connected component corresponding to v in (V, A_l). Because D is a minimal augmentation, e is necessary for feasibility. Since e is the only edge in $D \cap \delta(S)$ there must be some j such that either s_j or t_j is in S and the other is not in S. But then $T = \delta(S)$ would be in $\mathcal{V}_l$, and v would be red, which is a contradiction. □

Thus the algorithm in Figure 3 gives a 2-approximation algorithm for the generalized Steiner tree problem. The first 2-approximation algorithm for this problem was given by Agrawal, Klein, and Ravi [1]. Its use of the primal-dual method was made explicit by Goemans and Williamson [15].

3 Conclusions

Approximation algorithms for many NP-hard problems can be derived from the framework above: for example, network design problems (see the survey of Goemans and Williamson [16]), feedback vertex set problems [17, 5, 13], prize-collecting problems [15], multicut problems in trees [14], and others.

However, it is important to remember that the algorithm and analysis given above is only one potential way of applying the primal-dual technique, the one that developed historically from papers in the 80s and early 90s. A few recent papers have used their own variations of the primal-dual method. For instance, the paper of Jain and Vazirani [20] uses the same ideas for the uncapacitated facility location problem, but constructs a primal solution in a somewhat different way. Rajagopalan and Vazirani [24] use local search on top of a primal-dual algorithm to get an improved algorithm for the Steiner tree problem in some cases. Thus it is important to see that although many results can be derived directly from the algorithm in the preceeding section, it should be viewed as a starting point from which further modifications can be made to suit the problem at hand.

Acknowledgements

This primer is extracted from a much longer survey submitted to *Mathematical Programming* as part of the issue for the 17th International Symposium on Mathematical Programming. The author is grateful to the editors of *Mathematical Programming* for allowing him to make this extract available.

References

[1] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24:440–456, 1995.

[2] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.

[3] R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal on Computing*, 27:942–959, 1998.

[4] D. Bertsimas and C.-P. Teo. From valid inequalities to heuristics: A unified view of primal-dual approximation algorithms in covering problems. *Operations Research*, 46:503–514, 1998.

[5] F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson. A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs. *Operations Research Letters*, 22:111–118, 1998.

[6] V. Chvátal. *Linear Programming.* W.H. Freeman and Company, New York, NY, 1983.

[7] G. B. Dantzig, L. R. Ford, and D. R. Fulkerson. A primal-dual algorithm for linear programs. In H. W. Kuhn and A. W. Tucker, editors, *Linear Inequalities and Related Systems*, pages 171–181. Princeton University Press, Princeton, NJ, 1956.

[8] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[9] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71B:233–240, 1967.

[10] P. Erdős and L. Pósa. On the maximal number of disjoint circuits of a graph. *Publ. Math Debrecen*, 9:3–12, 1962.

[11] G. Even, J. S. Naor, B. Schieber, and L. Zosin. Approximating minimum subset feedback sets in undirected graphs with applications. *SIAM Journal on Discrete Mathematics*, 13:255–267, 2000.

[12] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[13] T. Fujito. Approximating node-deletion problems for matroidal properties. *Journal of Algorithms*, 31:211–227, 1999.

[14] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.

[15] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[16] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation algorithms for NP-hard problems.* PWS Publishing Company, 1997.

[17] M. X. Goemans and D. P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica*, 18:37–59, 1998.

[18] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556, 1982.

[19] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS Publishing Company, 1997.

[20] K. Jain and V. V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[21] P. Klein and R. Ravi. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *Proceedings of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, pages 39–55, 1993. Also appears as Brown University Technical Report CS-92-30.

[22] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[23] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[24] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, 1999.

[25] H. Saran, V. Vazirani, and N. Young. A primal-dual approach to approximation algorithms for network Steiner problems. In *Proceedings of Indo-US workshop on Cooperative Research in Computer Science*, pages 166–168, 1992.

[26] D. B. Shmoys. Computing near-optimal solutions to combinatorial optimization problems. In W. Cook, L. Lovász, and P. D. Seymour, editors, *Combinatorial Optimization*, pages 355–397. American Mathematical Society, 1995.

[27] G. Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, San Diego, CA, Third edition, 1988.

[28] C.-P. Teo. *Constructing approximation algorithms via linear programming relaxations: primal dual and randomized rounding techniques.* PhD thesis, MIT, 1996.

[29] V. V. Vazirani. *Approximation algorithms.* Springer, 2000.

[30] D. P. Williamson. *On the design of approximation algorithms for a class of graph problems.* PhD thesis, MIT, Cambridge, MA, September 1993. Also appears as Tech Report MIT/LCS/TR-584.

[31] D. P. Williamson. The primal-dual method for approximation algorithms. Submitted to *Mathematical Programming*, 2000.